Netflix Users Analysis Using Python

📝 Project Description:

Leveraging the power of Python and cutting-edge data analysis libraries, we delved into a fascinating dataset on Netflix users to uncover valuable insights. Explored key attributes such as Age, Gender, Subscription Plan, Monthly Revenue, Last Date of Activity, Join Date, and Device to gain a comprehensive understanding of user behavior and preferences. Employed advanced data visualization techniques to present findings in an insightful and visually appealing manner. Conducted in-depth analysis to identify trends, patterns, and correlations within the dataset, providing actionable insights for Netflix and related stakeholders.

Import Library

In [1]:
```
import pandas as pd
```

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
```

```
C:\Users\Syed Arif\anaconda3\lib\site-packages\scipy\__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected v
ersion 1.25.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Uploading Csv fle

In [3]:
```
df = pd.read_csv(r"C:\Users\Syed Arif\Desktop\Netflix User Base\Netflix Userbase.csv"
```

Data Preprocessing

.head()

head is used show to the By default = 5 rows in the dataset

In [4]:
```
df.head()
```

Out[4]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration
0	1	Basic	10	15-01-22	10-06-23	United States	28	Male	Smartphone	1 Month
1	2	Premium	15	05-09-21	22-06-23	Canada	35	Female	Tablet	1 Month
2	3	Standard	12	28-02-23	27-06-23	United Kingdom	42	Male	Smart TV	1 Month
3	4	Standard	12	10-07-22	26-06-23	Australia	51	Female	Laptop	1 Month
4	5	Basic	10	01-05-23	28-06-23	Germany	33	Male	Smartphone	1 Month

.tail()

tail is used to show last rows

In [5]:
```
df.tail()
```

Out[5]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration
2495	2496	Premium	14	25-07-22	12-07-23	Spain	28	Female	Smart TV	1 Month
2496	2497	Basic	15	04-08-22	14-07-23	Spain	33	Female	Smart TV	1 Month
2497	2498	Standard	12	09-08-22	15-07-23	United States	38	Male	Laptop	1 Month
2498	2499	Standard	13	12-08-22	12-07-23	Canada	48	Female	Tablet	1 Month
2499	2500	Basic	15	13-08-22	12-07-23	United States	35	Female	Smart TV	1 Month

.shape

It show the total no of rows & Column in the dataset

In [6]:
```
df.shape
```

Out[6]:
```
(2500, 10)
```

.Columns

It show the no of each Column

In [7]:
```
df.columns
```

Out[7]:
```
Index(['User ID', 'Subscription Type', 'Monthly Revenue', 'Join Date',
       'Last Payment Date', 'Country', 'Age', 'Gender', 'Device',
       'Plan Duration'],
      dtype='object')
```

.dtypes

This Attribute show the data type of each column

In [8]:
```
df.dtypes
```

Out[8]:
```
User ID              int64
Subscription Type    object
Monthly Revenue       int64
Join Date            object
Last Payment Date    object
Country              object
Age                   int64
Gender               object
Device               object
Plan Duration        object
dtype: object
```

.unique()

In a column, It show the unique value of specific column.

In [9]:
```
df["Country"].unique()
```

Out[9]:
```
array(['United States', 'Canada', 'United Kingdom', 'Australia',
       'Germany', 'France', 'Brazil', 'Mexico', 'Spain', 'Italy'],
      dtype=object)
```

.nuique()

It will show the total no of unque value from whole data frame

In [10]:
```
df.nunique()
```

Out[10]:
```
User ID              2500
Subscription Type       3
Monthly Revenue         6
Join Date             300
Last Payment Date      26
Country                10
Age                    26
Gender                  2
Device                  4
Plan Duration           1
dtype: int64
```

.describe()

It show the Count, mean , median etc

In [11]:
```
df.describe()
```

Out[11]:

	User ID	Monthly Revenue	Age
count	2500.00000	2500.000000	2500.000000
mean	1250.50000	12.508400	38.795600
std	721.83216	1.686851	7.171778
min	1.00000	10.000000	26.000000
25%	625.75000	11.000000	32.000000
50%	1250.50000	12.000000	39.000000
75%	1875.25000	14.000000	45.000000
max	2500.00000	15.000000	51.000000

.value_counts

It Shows all the unique values with their count

In [12]:
```
df["Country"].value_counts()
```

Out[12]:
```
United States     451
Spain             451
Canada            317
United Kingdom    183
Australia         183
Germany           183
France            183
Brazil            183
Mexico            183
Italy             183
Name: Country, dtype: int64
```

.isnull()

It shows the how many null values

In [13]:
```
df.isnull()
```

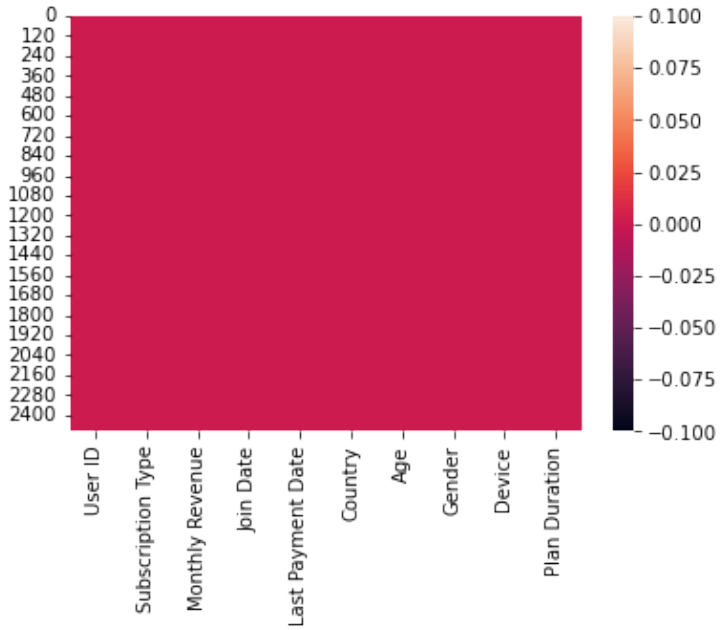
Out[13]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
2495	False	False	False	False	False	False	False	False	False	False
2496	False	False	False	False	False	False	False	False	False	False
2497	False	False	False	False	False	False	False	False	False	False
2498	False	False	False	False	False	False	False	False	False	False
2499	False	False	False	False	False	False	False	False	False	False

2500 rows × 10 columns

In [14]:
```
sns.heatmap(df.isnull())
```

Out[14]:
```
<AxesSubplot:>
```



In [15]:
```
df["Join Date"] = pd.to_datetime(df["Join Date"])
df["Last Payment Date"] = pd.to_datetime(df["Last Payment Date"])
```

In [16]:
```
import pandas as pd

# Assuming 'df' is your DataFrame with a 'Join Date' column
df['Join Date'] = pd.to_datetime(df['Join Date'])

# Extract month names
df['Join Month'] = df['Join Date'].dt.month_name()

# Display the DataFrame with the added 'Join Month' column
print(df)
```

```
      User ID Subscription Type  Monthly Revenue  Join Date Last Payment Date  \
0           1             Basic               10 2022-01-15        2023-10-06
1           2           Premium               15 2021-05-09        2023-06-22
2           3          Standard               12 2023-02-28        2023-06-27
3           4          Standard               12 2022-10-07        2023-06-26
4           5             Basic               10 2023-01-05        2023-06-28
...       ...               ...              ...        ...               ...
2495     2496           Premium               14 2022-07-25        2023-12-07
2496     2497             Basic               15 2022-04-08        2023-07-14
2497     2498          Standard               12 2022-09-08        2023-07-15
2498     2499          Standard               13 2022-12-08        2023-12-07
2499     2500             Basic               15 2022-08-13        2023-12-07

             Country  Age  Gender      Device Plan Duration Join Month
0      United States   28    Male  Smartphone       1 Month    January
1             Canada   35  Female      Tablet       1 Month        May
2     United Kingdom   42    Male    Smart TV       1 Month   February
3          Australia   51  Female      Laptop       1 Month    October
4            Germany   33    Male  Smartphone       1 Month    January
...              ...  ...     ...         ...           ...        ...
2495           Spain   28  Female    Smart TV       1 Month       July
2496           Spain   33  Female    Smart TV       1 Month      April
2497   United States   38    Male      Laptop       1 Month  September
2498          Canada   48  Female      Tablet       1 Month   December
2499   United States   35  Female    Smart TV       1 Month     August

[2500 rows x 11 columns]
```

Why we Use (get_continent) in Python:

This library can help you find the continent of a given country

In [17]:
```python
# Deriving some useful features using lambda function

def get_continent(country):
    """returns the continent of the given country"""

    if country in {"United States", "Canada", "Mexico"}:
        return "North America"
    if country in {"France", "Germany", "United Kingdom", "Italy", "Spain"}:
        return "Europe"
    if country == "Brazil":
        return "South America"
    if country == "Australia":
        return "Australia"
    return "Africa / Asia"


def get_age_class(age):
    """returns the age class of a given age"""

    return "Kid" if age < 11 \
    else "Teen" if age < 20  \
    else "Young" if age < 40 \
    else "Senior" if age < 70 \
    else "Elderly"
df["Country"] = df["Country"].apply(lambda x: get_continent(x))
df["Age"] = df["Age"].apply(lambda x : get_age_class(x))
```

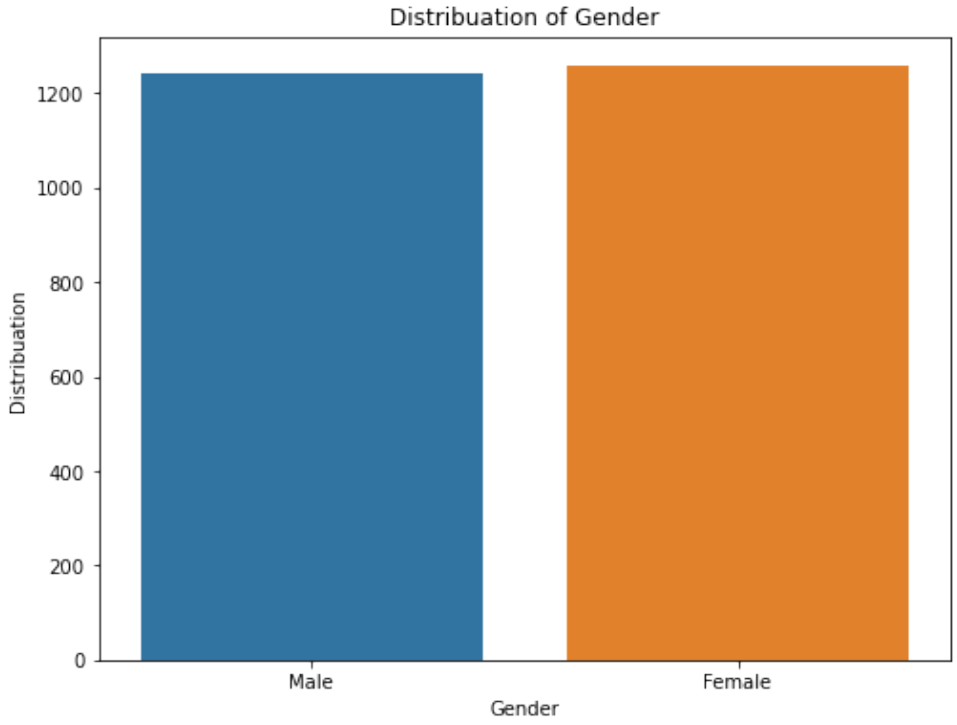
In [18]:
```
df
```

Out[18]:

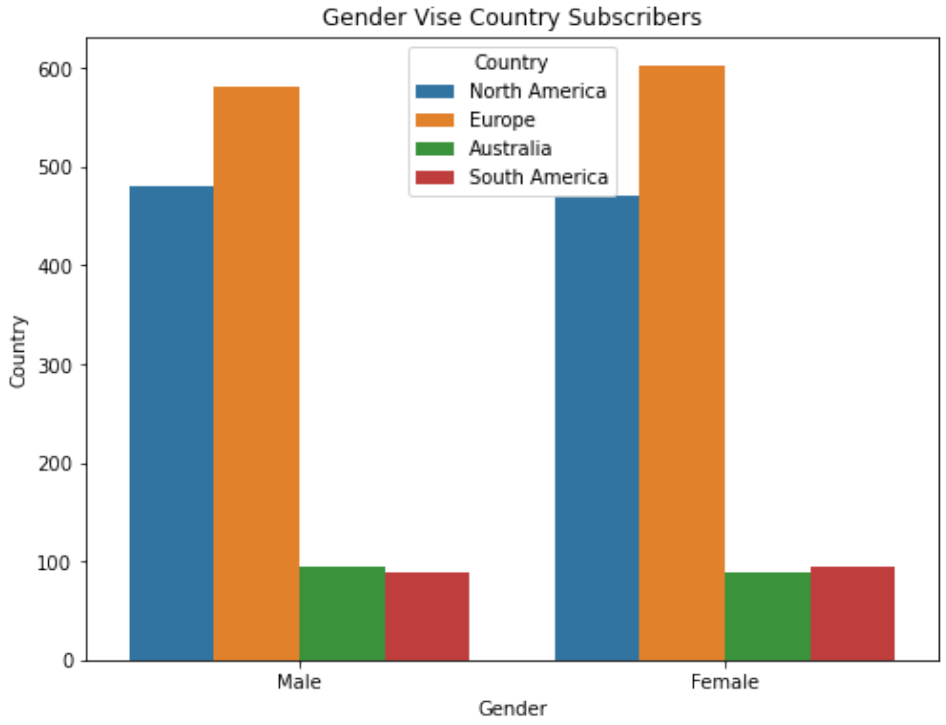
	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration	
0	1	Basic	10	2022-01-15	2023-10-06	North America	Young	Male	Smartphone	1 Month	J
1	2	Premium	15	2021-05-09	2023-06-22	North America	Young	Female	Tablet	1 Month	
2	3	Standard	12	2023-02-28	2023-06-27	Europe	Senior	Male	Smart TV	1 Month	Fe
3	4	Standard	12	2022-10-07	2023-06-26	Australia	Senior	Female	Laptop	1 Month	C
4	5	Basic	10	2023-01-05	2023-06-28	Europe	Young	Male	Smartphone	1 Month	J
...	...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14	2022-07-25	2023-12-07	Europe	Young	Female	Smart TV	1 Month	
2496	2497	Basic	15	2022-04-08	2023-07-14	Europe	Young	Female	Smart TV	1 Month	
2497	2498	Standard	12	2022-09-08	2023-07-15	North America	Young	Male	Laptop	1 Month	Sept
2498	2499	Standard	13	2022-12-08	2023-12-07	North America	Senior	Female	Tablet	1 Month	Dec
2499	2500	Basic	15	2022-08-13	2023-12-07	North America	Young	Female	Smart TV	1 Month	

2500 rows × 11 columns

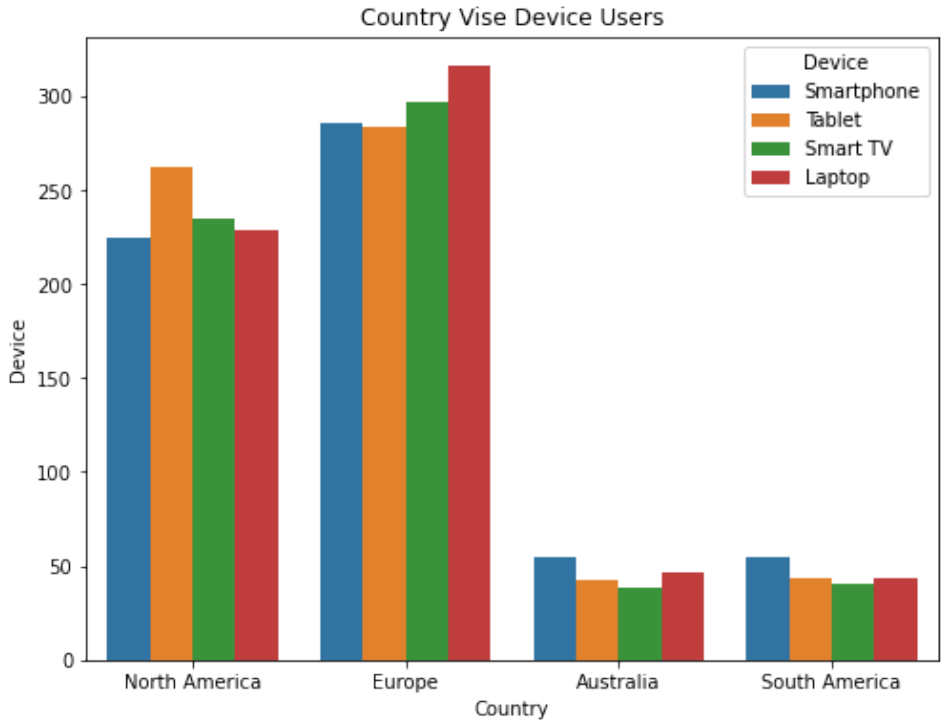
In [19]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Gender')
plt.xlabel('Gender')
plt.ylabel('Distribuation')
plt.title('Distribuation of Gender')
plt.show()
```



In [20]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Gender', hue ="Country")
plt.xlabel('Gender')
plt.ylabel('Country')
plt.title('Gender Vise Country Subscribers')
plt.show()
```

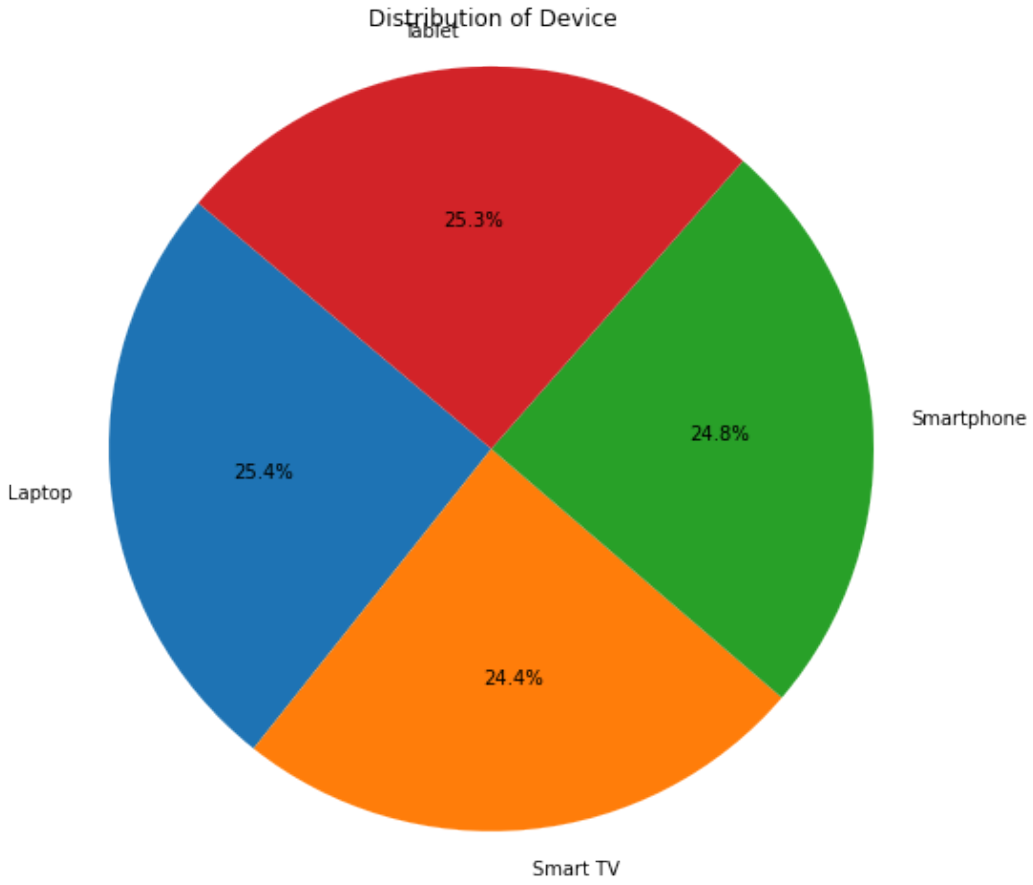


In [21]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Country', hue ="Device")
plt.xlabel('Country')
plt.ylabel('Device')
plt.title('Country Vise Device Users')
plt.show()
```

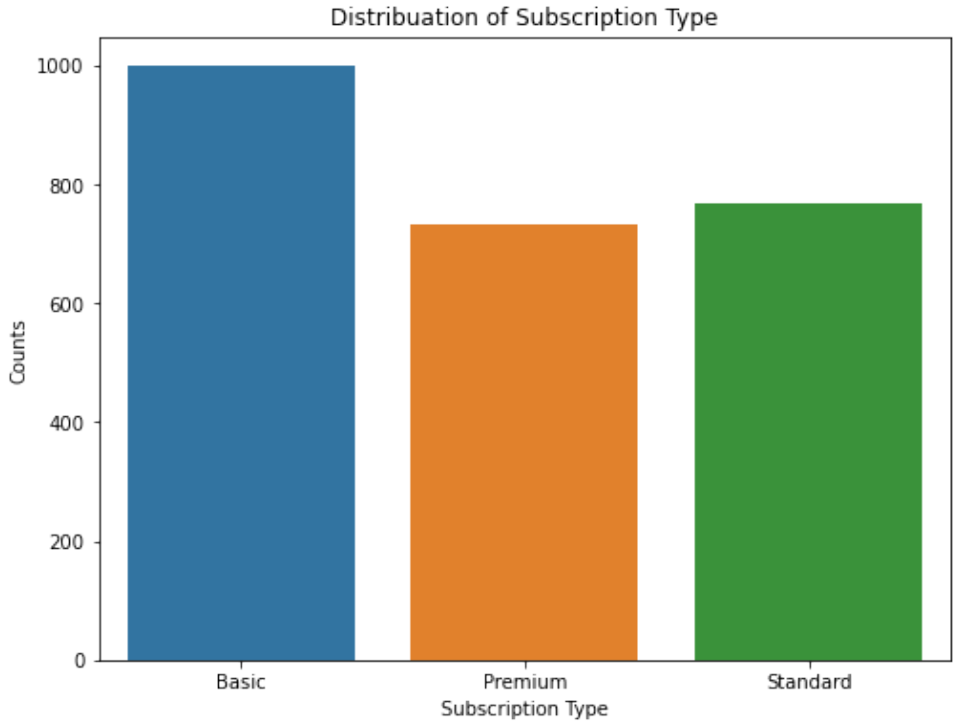


In [22]:
```
# Group the data by Feedback and calculate the count of each category
Device = df.groupby('Device').size()

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(Device, labels=Device.index, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Device')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



In [23]:
```
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Subscription Type')
plt.xlabel('Subscription Type')
plt.ylabel('Counts')
plt.title('Distribuation of Subscription Type')
plt.show()
```



In [26]:
```
df
```

Out[26]:

	User ID	Subscription Type	Monthly Revenue	Join Date	Last Payment Date	Country	Age	Gender	Device	Plan Duration	
0	1	Basic	10	2022-01-15	2023-10-06	North America	Young	Male	Smartphone	1 Month	J
1	2	Premium	15	2021-05-09	2023-06-22	North America	Young	Female	Tablet	1 Month	
2	3	Standard	12	2023-02-28	2023-06-27	Europe	Senior	Male	Smart TV	1 Month	Fe
3	4	Standard	12	2022-10-07	2023-06-26	Australia	Senior	Female	Laptop	1 Month	C
4	5	Basic	10	2023-01-05	2023-06-28	Europe	Young	Male	Smartphone	1 Month	J
...	...	...	...	...	...	...	...	...	...	...	
2495	2496	Premium	14	2022-07-25	2023-12-07	Europe	Young	Female	Smart TV	1 Month	
2496	2497	Basic	15	2022-04-08	2023-07-14	Europe	Young	Female	Smart TV	1 Month	
2497	2498	Standard	12	2022-09-08	2023-07-15	North America	Young	Male	Laptop	1 Month	Sept
2498	2499	Standard	13	2022-12-08	2023-12-07	North America	Senior	Female	Tablet	1 Month	Dec
2499	2500	Basic	15	2022-08-13	2023-12-07	North America	Young	Female	Smart TV	1 Month	

2500 rows × 11 columns

In [27]:
```
Joining_Months_Counts = df['Join Month'].value_counts()
Joining_Months_Counts.plot(kind='bar')
plt.xlabel('Join Month')
plt.ylabel('Count')
plt.title('Joining Counts By Months')
```

Out[27]: Text(0.5, 1.0, 'Joining Counts By Months')